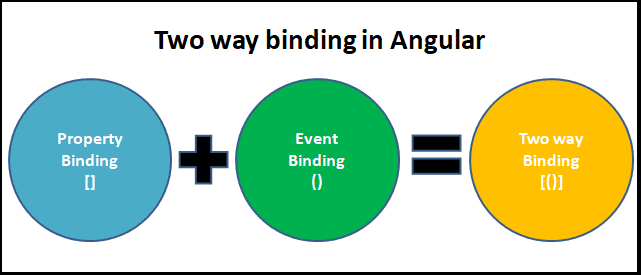
**1. Data Binding:-** Data binding is a technique, where the data stays in sync between the component/Model and the view. Whenever the user updates the data in the view, Angular updates the component. When the component gets new data, the Angular updates the view. The data binding in Angular can be broadly classified into two groups. One way binding or two-way binding.

1.1.One-Way Binding:- In one way binding data flows from one direction. Either from view to component or from component to view.

1.1.1.From Component to View:- To bind data from component to view, we make use of either String Interpolation or Property Binding.

1.1.2. From view to component:- To bind data from view to component, we make use of Event binding.

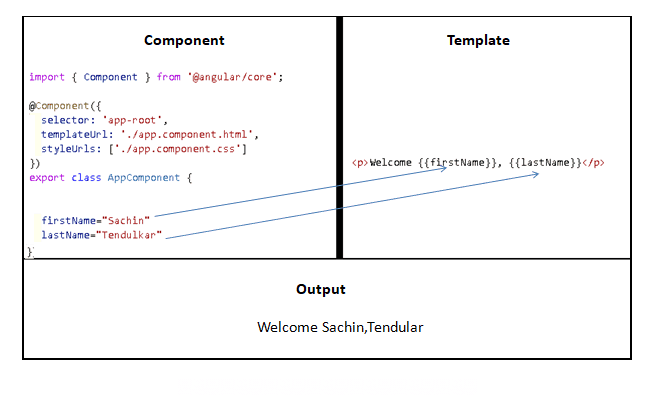
1.2.Two-way Binding:- Two-way binding means that changes made to our model in the component are propagated to the view and that any changes made in the view are immediately updated in the underlying component. Two-way binding is useful in data entry forms.



**2. Interpolation:** Angular interpolation is also known as “string interpolation”. Interpolation allows us to include template-expressions as part of any string literal. You can use interpolation wherever you use a string literal in the view.

Syntax:-{{template-Expression}}

The content inside the double braces is called Template Expression. The Angular first evaluates the Template Expression and converts it into a string. Then it replaces Template expression with the result in the original string in the HTML. Whenever the template expression changes, the Angular updates the original string again.



Note:-

* The interpolation must only be written in string literal.
* The template Expression may be ‘component function or built-in function’.
* we can perform the concatenation between two templateExpressions.
* {{‘sukumar’+’atmakuru’}}
* Simple mathematical operation can also be written as template expression.
* {{20+30}}
* we bind interpolation to html element property.
* Example:- <p style.color={{giveMeRed}}>This is red</p>
* NgNonBindable: Use ngNonBindable to tell Angular not to compile or bind the contents of the current DOM element. I.e any expression is not evaluated but shown as it is.

Example:

1.app.component.html

<div style.color={{prop}}>

  <h5>Name:{{name}}</h5>

  <h5>Age:{{age}}</h5>

  <h5>Pass:{{pass}}</h5>

  <h5>Qaul:{{display()}}</h5>

</div>

**2.**app.component.css

div{

    border:2px solid red;

    background-color: black;

}

h5{

    font-size:20px;

}

3.app.component.ts

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'first-app';

  name:string='sukumar';

  age:number=41;

  pass:boolean=true;

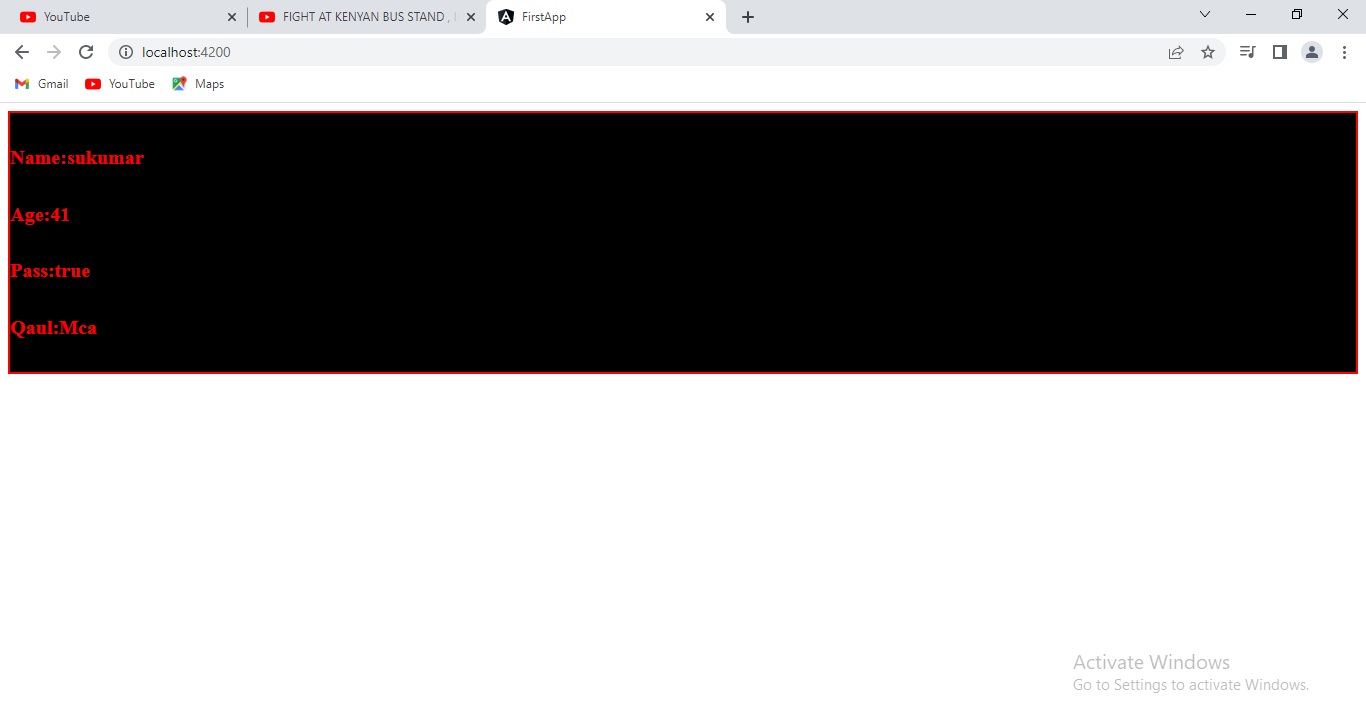
  display(){

    return 'Mca';

  }

  prop:string='red';

}



**3. property Binding:**

**1.**app.component.html:

<button style="margin-top:5px; font-size:20px" disabled={{dis}}>Click </button>

2.app.component.cs

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

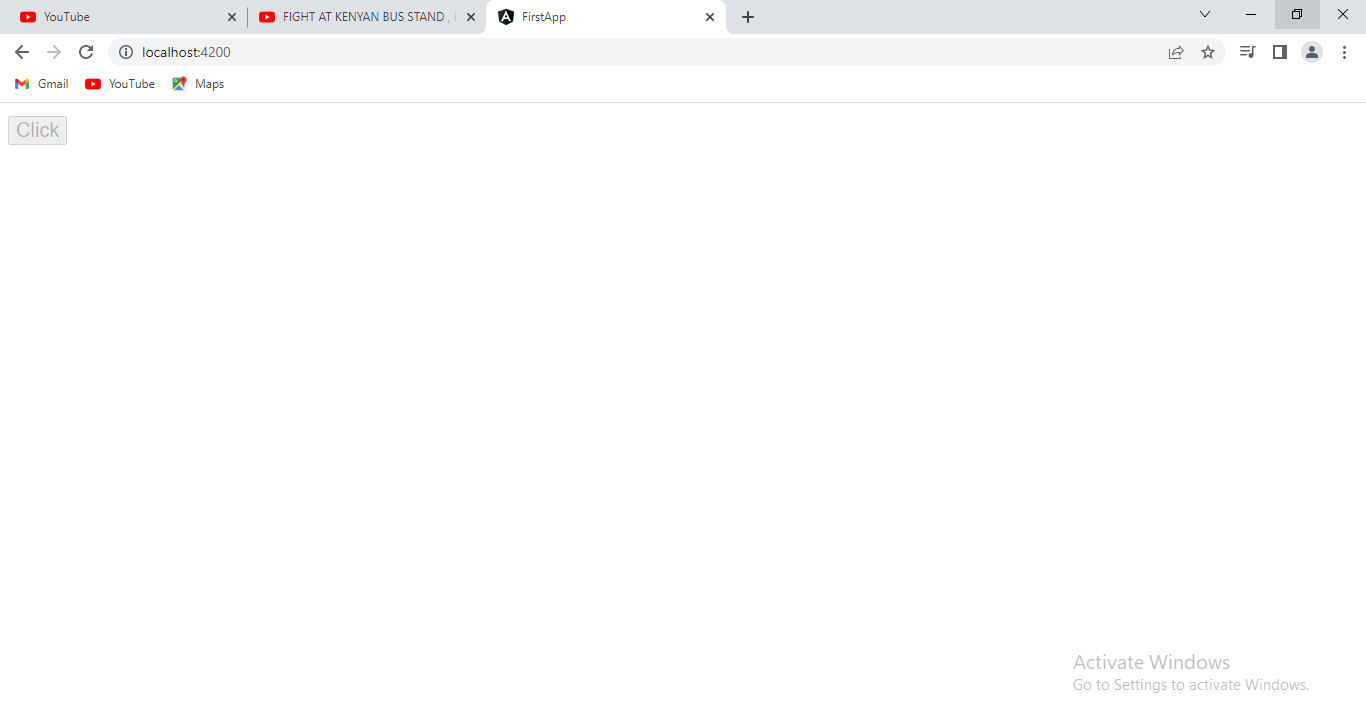
export class AppComponent {

  title = 'first-app';

  dis:boolean=false;

}

Output:-



Problem with interpolation:

Some element attributes accept only either true or false like disabled,..etc. If you assign value to them by interpolation, then any value which is returned by interpolation is **string type** and will be considered as true.

Therefore in above output, button was in disabled state. To overcome this problem, property binding has been introduced.

Syntax:

[binding-target]=’bingding-source’;

* The binding-target is attribute name of html element. It should be in-between [ and ].
* The binding-source is property name of component.

Note: When the component values change, the Angular updates the view. But if the values changes in the view, the Angular does not update the component.

Example:

1.app.component.html

<h3 [id]='sv'>name:{{name}}</h3>

<h3 [id]="sv">Age:{{age}}</h3>

<h3 [id]="sv">Pass:{{pass}}</h3>

<button style="margin-top:5px; font-size:20px" [disabled]='dis'>Click </button>

2.app.component.css

#rock{

    font-size:30px;

    color:green;

}

3.app.component.ts

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'first-app';

  name:string='sukumar';

  age:number=41;

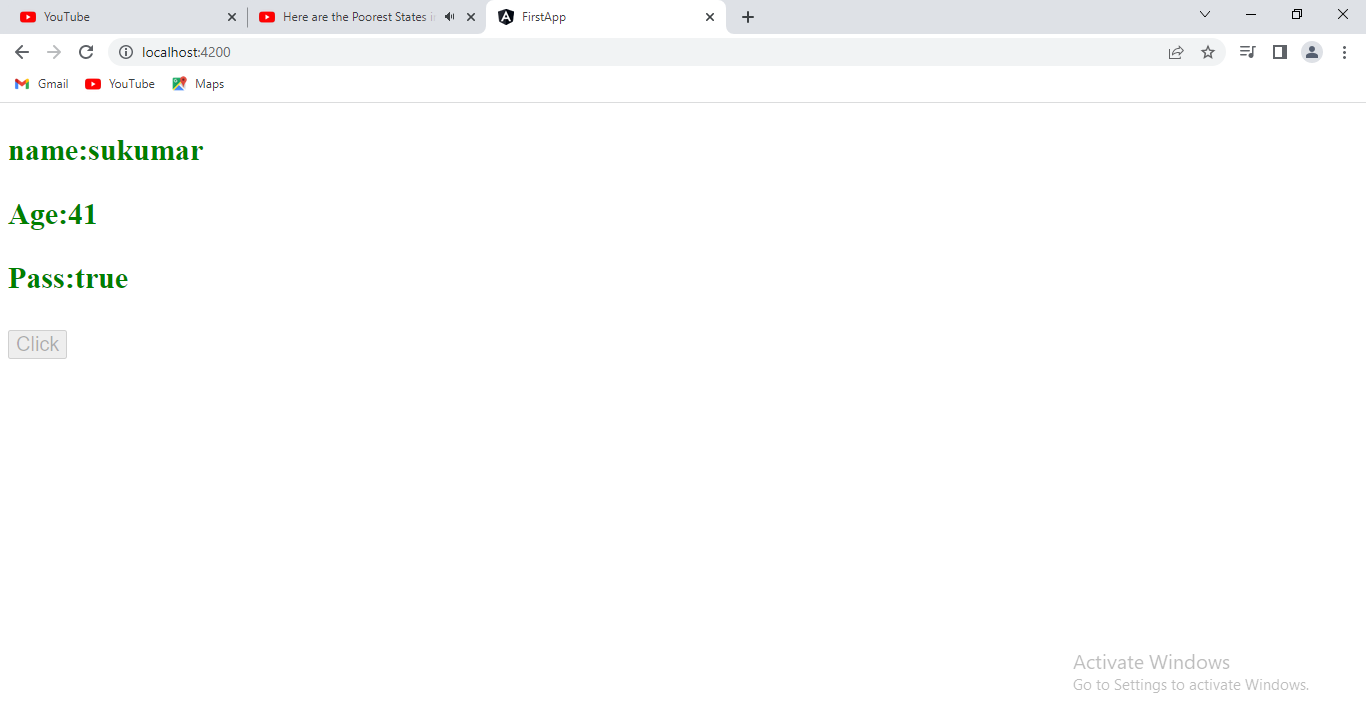
  pass:boolean=true;

  dis:boolean=true;

  sv:string='rock';

}

Output:



**4.class Binding:**

**4.1.**Conditional Based class Binding:

**MODEL:1**

* Write a css rule in component.css file.

* Declare another property in component. Ts file. That property must be boolean type.

If it contains true, the styles will be applied to html element.

If it contains false, the styles will not be applied to html element.

* Open the .html file and bind class name to class attribute of element.

Syntax:-

<tag [class. Name of class]=’propertyname’>content</tag>

Example:

1.app.component.html

<div [class.apply]='valid'>

    This Example

</div>

2. app.component.css

.apply

{

  background-color: yellow;

  font-size: 20px;

  border: 1px solid red;

}

3.app.component.ts

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

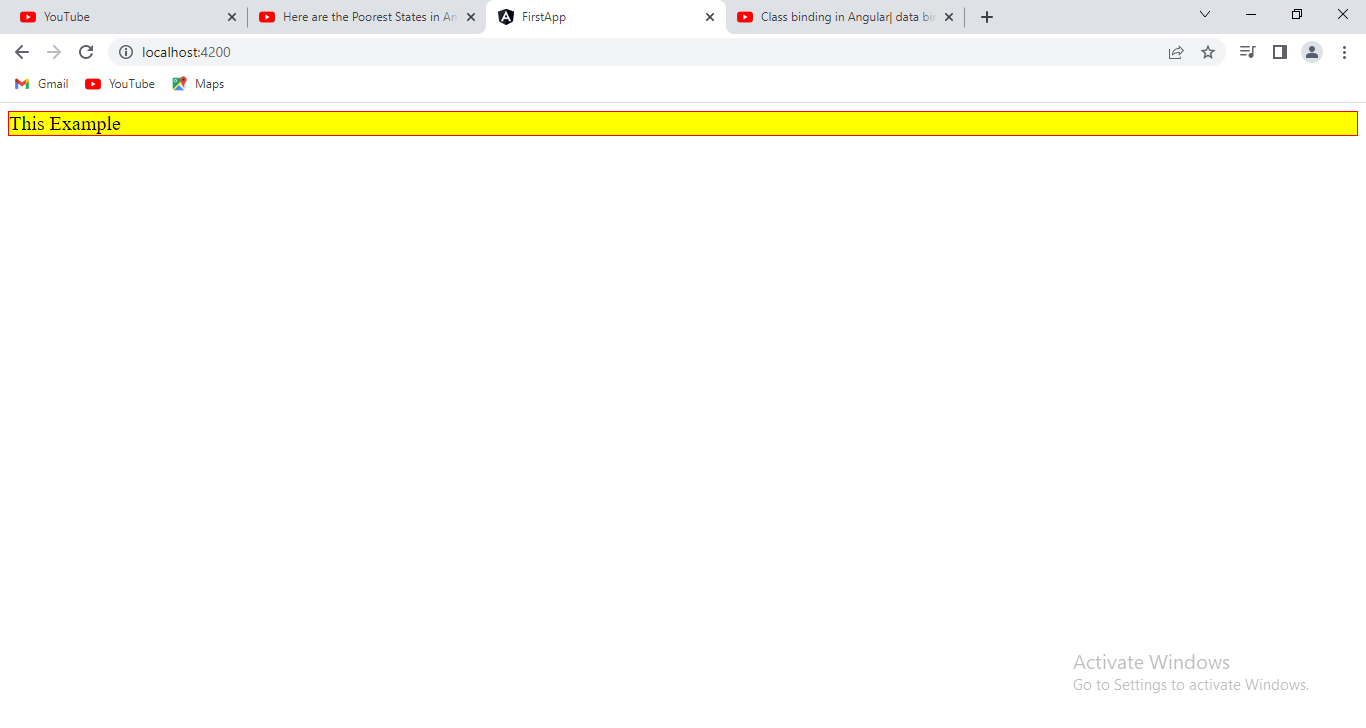
export class AppComponent {

  title = 'first-app';

  valid:boolean=true;

}

Output:-



MODEL:2

* Write a two css rules in component.css file.

* Declare another property in component. Ts file. That property must be boolean type.

If it contains true, the first style will be applied to html element.

If it contains false, the second style will be applied to html element.

* Open the .html file and bind class name to class attribute of element.

Syntax1:-

<tag [class]=”proper-name?’cssstyle-1’:’cssstyle-2’”>content</tag>

Example:

1.app.component.html

<div [class]="valid?'active':'inactive'">

    This Example

</div>

2.app.component.css

.active

{

  background-color: yellow;

  font-size: 20px;

  border: 1px solid red;

}

.inactive

{

  background-color: red;

  font-size: 20px;

  border: 1px solid black;

}

3.app.component.ts

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

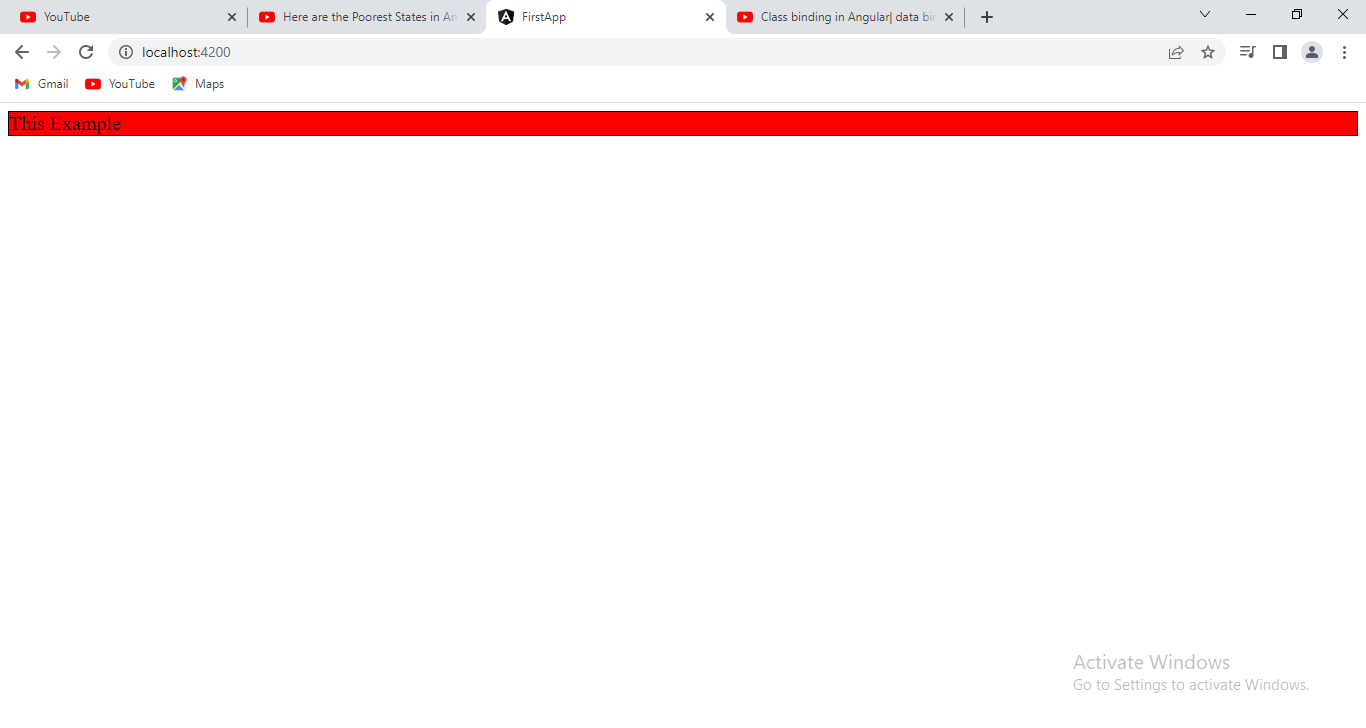
export class AppComponent {

  title = 'first-app';

  valid:boolean=false;

}

Output:



**Model:3**

* Write a two css rules in component.css file.

* Declare two properties in component. Ts file. That properties must be boolean type.

If first property contains true, the first style will be applied to html element.

If second property contains true, the second style will be applied to html element.

* Open the .html file and bind class name to class attribute of element.

Syntax:

<tag[class]=”{cssstyle-1:property-1,cssstyle-2:property-2}”>content</tag>

1.app.component.hmtl

<div [class]="{active:valid,inactive:valid1}">

    This Example

</div>

2.app.component.css

.active

{

  background-color: yellow;

  font-size: 20px;

  border: 1px solid red;

}

.inactive

{

  background-color: red;

  font-size: 20px;

  border: 1px solid black;

}

3.app.component.ts

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

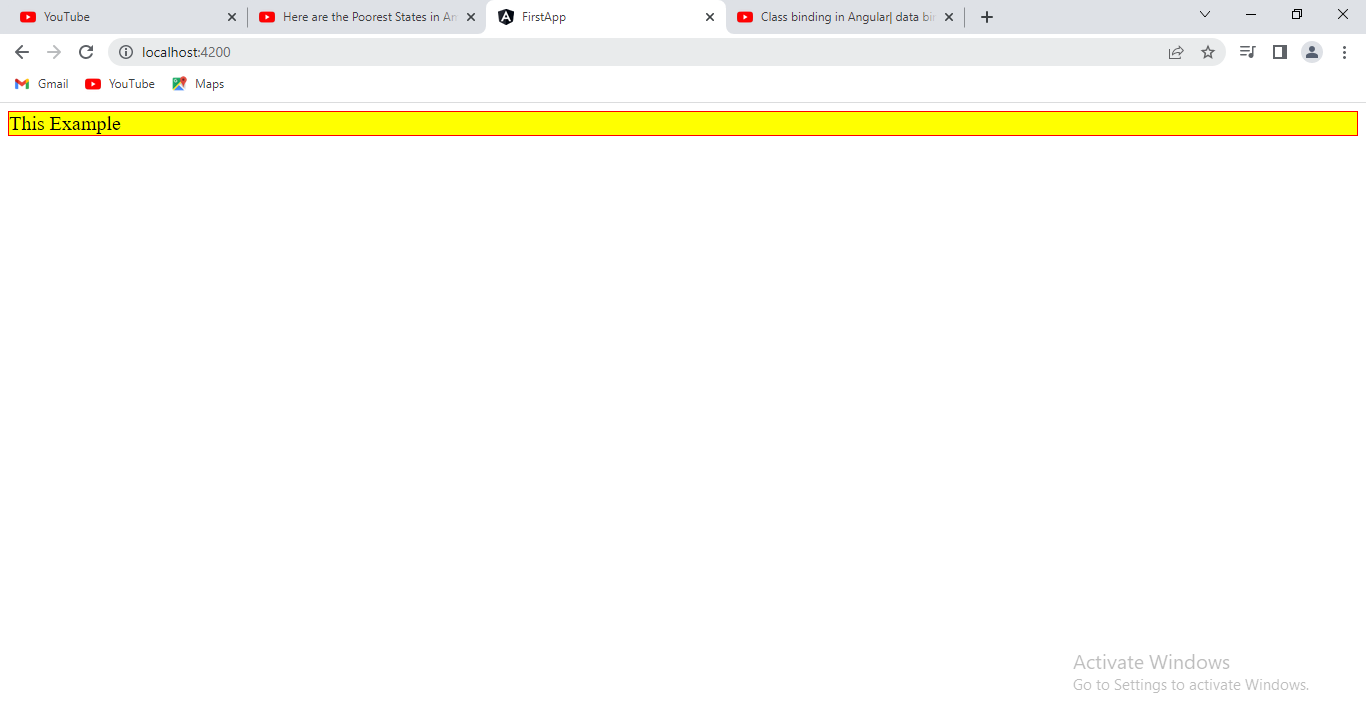
  title = 'first-app';

  valid:boolean=true;

  valid1:boolean=false;

}

Output:



Example:

App.component.html

<button (click)="abc()">OkDarling</button>

<p>Value Is:{{title}}</p>

<h2 [class]="[x,y]">Sukumar</h2>

App.component.css

.a{

    color:red;

    font-size: 30px;

}

.b

{

    border:1px solid green;

}

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'first-app';

  abc()

  {

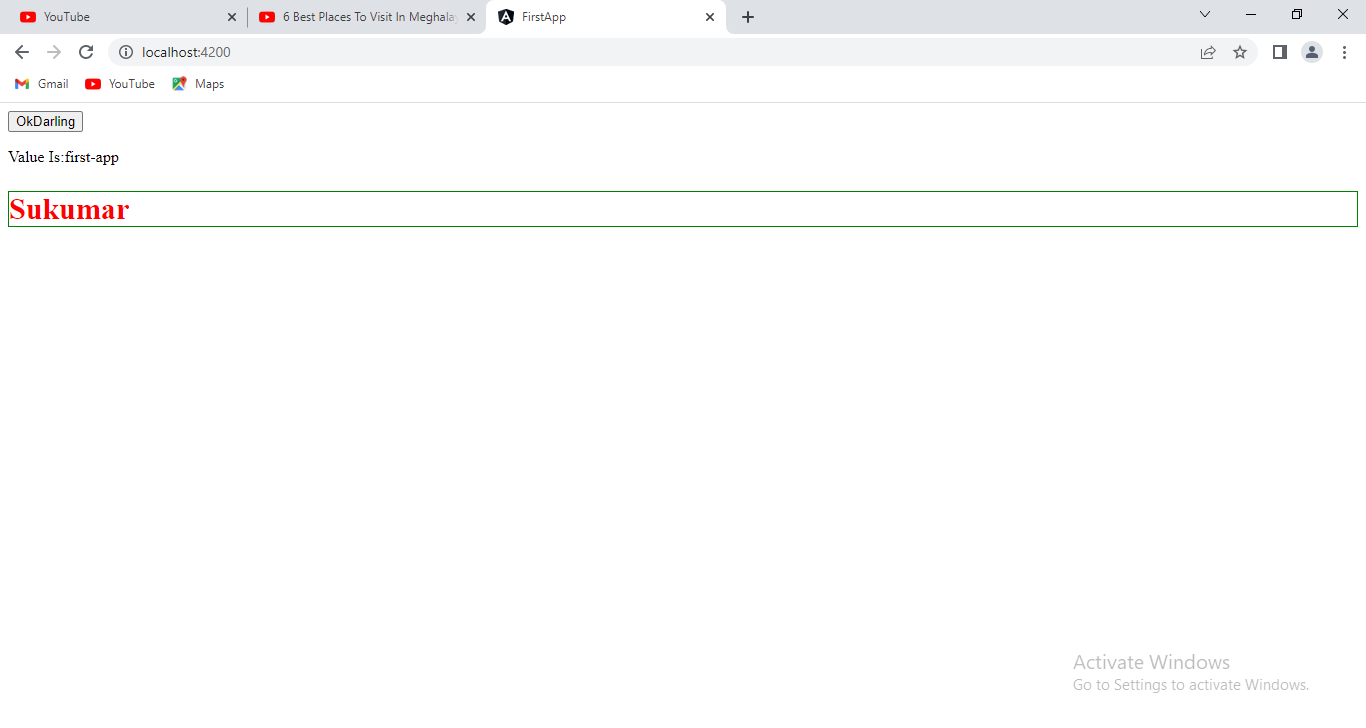
    this.title="he clicked me";

  }

  x:string='a';

  y:string='b';

}



**5. Style Binding:** We can set the inline styles of an HTML element using the style binding in angular. You can also add styles conditionally to an element, hence creating a dynamically styled element.

Syntax:1

<tag [style.propertyname]=’component-class propertyname’>content</tag>

Syntax:2

<tag [style]=’component-class propertyname’>content</tag>

Example:

1.app.component.html

<div style={{one}}>Element-1</div  >

<div [style]='two' >Element-2</div  >

<div [style.color]="three" >Element-3</div  >

<div [style]="four" >Element-3</div  >

2.app.component.ts

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'first-app';

  one:string='color:red;border:2px solid green;margin:5px;'

  two:string='color:blue;border:1px solid orange;margin:5px;'

  three:string='green'

  four:object={

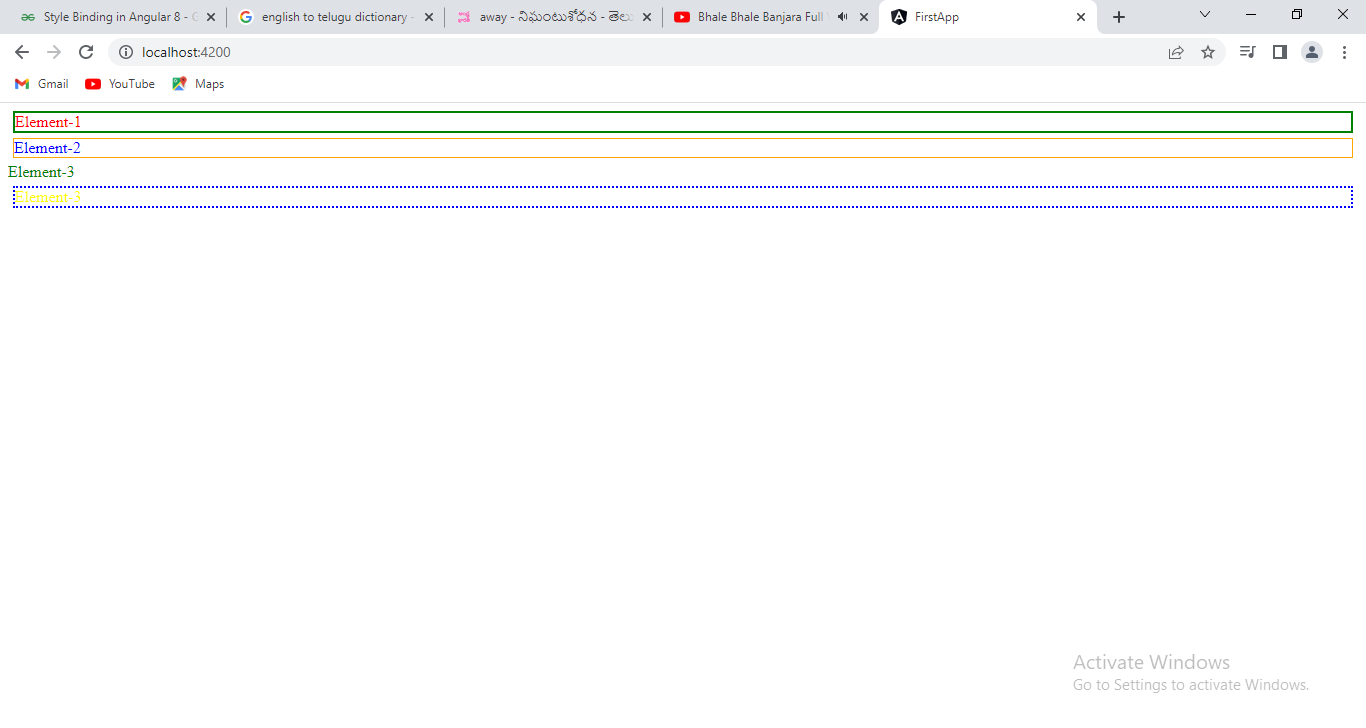
    color:'yellow',

    border:'2px dotted blue',

    margin:'5px'

  }

}



**6. Event Binding:** Event binding allows us to bind events such as keystroke, clicks, hover, touche, etc to a method in component. It is one way from view to component.

Syntax-1: (event-name)=’template statement’.

Syntax-2 :on-eventname=’template statement’.

The template statement is method name which exists in component.

6.1.EventPayLoad:- when event occurs, The event carries event pay load from template to method/event handler in component.

Syntax:

(event-name)=method-name($event)---🡪 this should be written in template.

Method-name(varname:Event)

{}

Example:

1.app.component.html:

<button (click)="abc()">OkDarling</button>

<p>Value Is:{{title}}</p>

2. app.component.ts:

import { Component } from '@angular/core';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'first-app';

  abc()

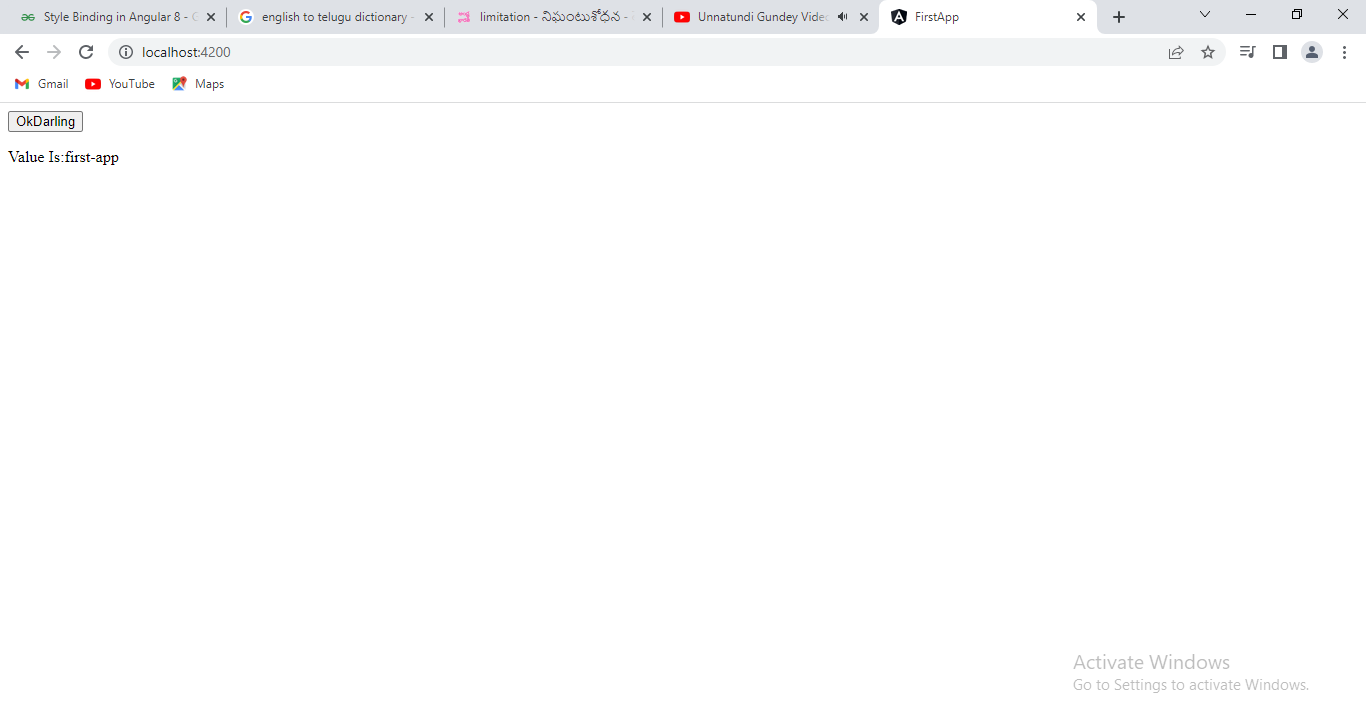
  {

    this.title="he clicked me";

  }

}

**Output:**



After Clicking the button.

